#SHAREorg

SHARE
Technology · Connections · Results

Simple BCPii Programming for the z/OS System Programmer

Steve Warren

IBM Corporation

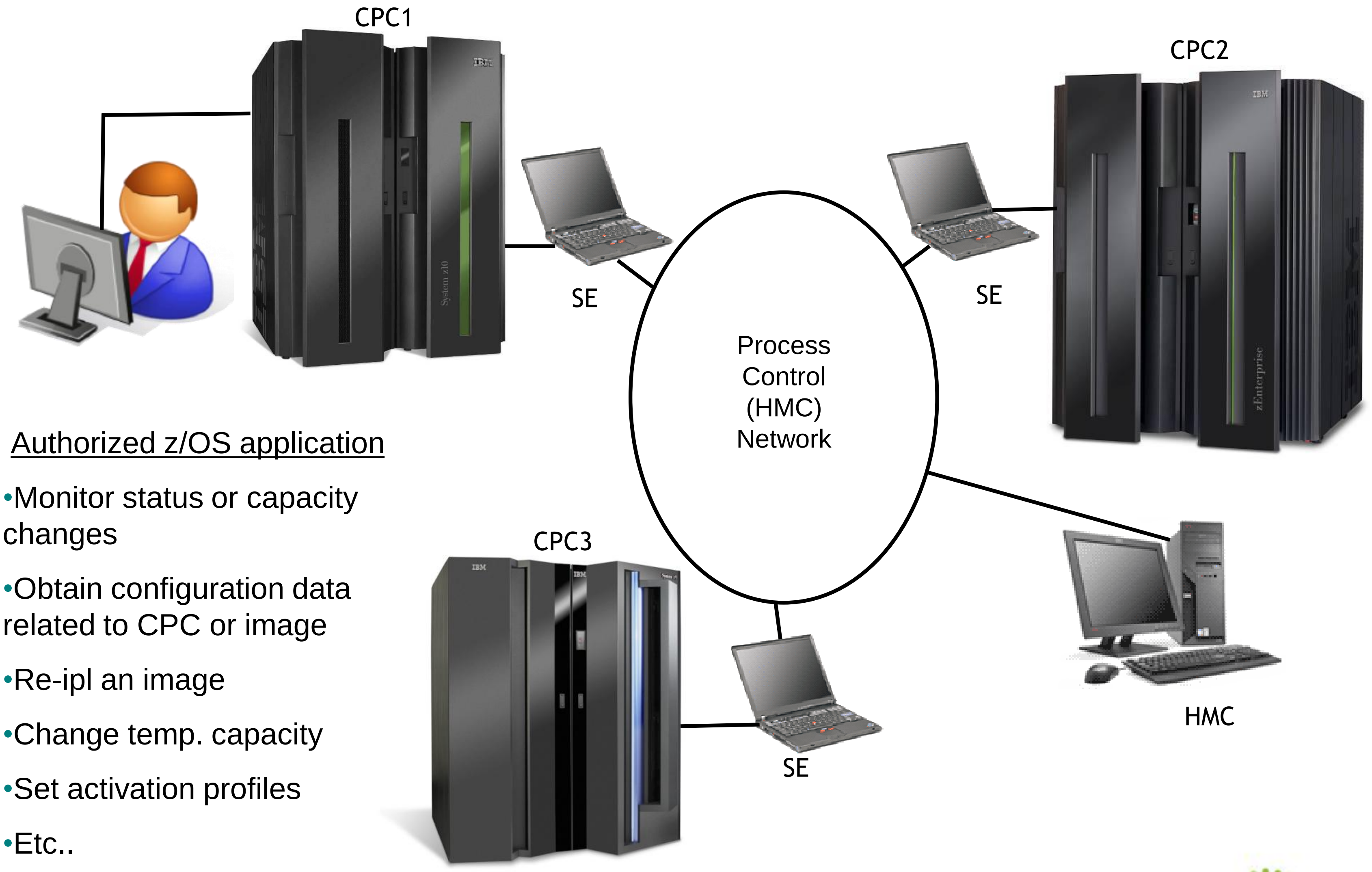
Wednesday, March 14, 3:00PM
Session Number 10626

Trademarks

Agenda

- BCPii quick overview
- BCPii installation overview
- Quick BCPii z/OS 1.13 Update
- BCPii Programming 101
  - Programming Environment
  - Language support
  - Let’s meet the APIs
  - Basic programming example
- BCPii Programming kicked up a notch
  - Dynamically adapting to the current HMC config
  - Dealing with communication outages
  - Dealing with BCPii outages
  - Debugging Programming Errors

What is BCPii?



Authorized z/OS application

- Monitor status or capacity changes
- Obtain configuration data related to CPC or image
- Re-ipl an image
- Change temp. capacity
- Set activation profiles
- Etc..

What is BCPii?

- Base Control Program internal interface
  - Allows authorized z/OS applications to have HMC-like control over systems in the process control (HMC) network
  - A set of authorized APIs provided
- Complete communication isolation of existing networks (intranet/internet) from the process control (HMC network)
  - BCPii communication is completely within the base z/OS
- A z/OS address space to manage authorized interaction with the interconnected hardware

What is z/OS BCPii vs. BCPii mentioned in TSA?

- Tivoli System Automation (ProcOps) allows its automation product to use one of 2 transport protocols:
  - SNMP over an IP network
  - BCPii protocol (internal transport)
- TSA's BCPii implementation is similar but not z/OS BCPii and requires TSA, Netview and Comm Server.
- BCPii transport in TSA is for TSA usage only
- z/OS BCPii APIs can be invoked from ANY address and has no other product requirements.
- GDPS is one of the main exploiters of TSA.

Who uses BCPii?

- z/OS operating system components
  - System Status Detection (SSD) from Sysplex Failure Manager (SFM)
  - Capacity Provisioning Manager (CPM)
  - Hardware Configuration Definition (HCD)
- Vendor applications
  - Control center, system management applications
  - Several likely to go GA soon.
- In-house (customer-written) applications

Quick BCPii Installation Overview

- Configure the local SE to support BCPii
  - HMC/SE administrator
- Authorize an application to use BCPii
  - Security administrator
- Configure the address space
  - z/OS System Administrator
- Set up the event notification mechanism for z/OS UNIX callers (if required)
  - z/OS System Administrator and Security Administrator

Quick BCPii Installation Overview – Msgs before Installation

```
IRR813I NO PROFILE WAS FOUND IN THE STARTED CLASS FOR 497
        HWIBCPII WITH JOBNAME HWIBCPII. RACF WILL USE ICHRIN03.
IEF196I         2 //HWIBCPII EXEC IEESYSAS,PROG=HWIAMIN2
IEF196I  STMT NO. MESSAGE
IEF196I           XX* THE IEESYSAS PROCEDURE IS SPECIFIED IN THE
IEF196I                   00150000
IEF196I           XX* PARAMETER LIST TO IEEMB881 BY MVS COMPONENTS
IEF196I                   00200000
IEF196I         2 IEFC001I PROCEDURE IEESYSAS WAS EXPANDED USING SYSTEM
IEE252I MEMBER CTIHWI00 FOUND IN SYS1.PARMLIB
HWI007I BCPII IS ATTEMPTING COMMUNICATION WITH THE LOCAL CENTRAL 965
PROCESSOR COMPLEX (CPC).
HWI022I THE SNMP COMMUNITY NAME REQUIRED BY BCPII FOR THE LOCAL  536
CPC WAS REJECTED BY THE SUPPORT ELEMENT.  CORRECT THE COMMUNITY NAME
IN THE SECURITY PROFILE THAT WAS RETRIEVED BY BCPII USING THIS ENTITY
(HWI.TARGET.IBM390PS.R132).
IEA989I SLIP TRAP ID=X13E MATCHED.  JOBNAME=HWIBCPII, ASID=001A.
HWIBCPII HWIBCPII IEFPROC   00:00:00      2,117     00
IEF404I IEESYSAS - ENDED - TIME=13.31.03
IEF196I IEF142I IEESYSAS HWIBCPII - STEP WAS EXECUTED - COND CODE 0000
IEF196I IEF376I  JOB/HWIBCPII/STOP  2009239.1331 CPU    0MIN 00.02SEC
IEF196I SRB    0MIN 00.00SEC
IEF352I ADDRESS SPACE UNAVAILABLE
IEF196I IEF352I ADDRESS SPACE UNAVAILABLE
IEA989I SLIP TRAP ID=X33E MATCHED.  JOBNAME=*UNAVAIL, ASID=001A.
HWI006I BCPII ADDRESS SPACE HAS ENDED.
IXC104I SYSTEM STATUS DETECTION PARTITIONING PROTOCOL ELIGIBILITY: 399
  SYSTEM CANNOT TARGET OTHER SYSTEMS.
    REASON: BCPII SERVICES NOT AVAILABLE
  SYSTEM IS NOT ELIGIBLE TO BE TARGETED BY OTHER SYSTEMS.
    REASON: BCPII SERVICES NOT AVAILABLE
IXC107E SYSTEM STATUS DETECTION PARTITIONING PROTOCOL CONFIGURATION IS NOT COMPLETE
```

Quick BCPii Installation Overview – Configuring the local SE

- Make sure microcode is at the minimum level
- Ensure BCPii communications on the SE
  - Turn on *“Cross Partition Authority”*
- Define the BCPii community name on the SE
  - *Turn on “Enable SNMP APIs”*
  - *Add a BCPii community name for the BCPii port*
- Repeat these same steps above for any additional CPCs where communication is desired

Quick BCPii Installation Overview – Authorizing the application

- Give program authority to application issuing BCPii APIs
- Give general SAF authority
  - Give at least read authority to the FACILITY class profile HWI.APPLNAME.HWISERV
- Give SAF authority to a specific resource
  - Give proper authority to the specific SAF profile for the entity you wish to protect
  - CPC, activation profile and user-defined image group connections
    - HWI.TARGET.netid.nau
  - Image connections
    - HWI.TARGET.netid.nau.imagename
  - Capacity record connections
    - HWI.CAPREC.netid.nau.caprec

Quick BCPii Installation Overview – Authorizing BCPii to CPC

- Specify the community name on the CPC FACILITY class definition
  - Define community name in SAF for CPC
    - Associate the community name already defined on the support elemente for BCPii with the SAF CPC profile already defined using the APPLDATA field
  - Can be done at the time of the CPC definition
  - BCPii propagates this community name to the SE when a connection is established with the SE.
- *NOTE: the local CPC's community name MUST be defined to SAF or the BCPii address space will not come up.*

Quick BCPii Installation Overview – BCPii address space config

- Make sure the *high-level-qualifier.SCEERUN2* and *high-level-qualifier.SCEERUN* data sets are in the link list concatenation.
- Optionally, configure the BCPii CTRACE parmlib member (CTIHWI00) to set up default CTRACE options for the address space.

Quick BCPii Installation Overview – z/OS Unix Event Authorization

- Optionally, set up the event notification mechanism for z/OS UNIX callers
  - Start the Common Event Adapter address space in full-function mode.
  - Grant a z/OS UNIX BCPii application authority to listen to ENF68 events using profile definitions in the SAF SERVAUTH

Quick BCPii Installation Overview – Msgs after Successful Installation

```
IEE252I MEMBER CTIHWI00 FOUND IN SYS1.PARMLIB
HWI016I THE BCPII COMMUNICATION RECOVERY ENVIRONMENT IS 225
NOW ESTABLISHED.
HWI007I BCPII IS ATTEMPTING COMMUNICATION WITH THE LOCAL CENTRAL 226
PROCESSOR COMPLEX (CPC).
HWI001I BCPII IS ACTIVE.
IXC111I LOGICAL PARTITION REMOTE CONNECTION INFORMATION 048
  CONNECTION STATUS:     CONNECTED
  SYSTEM NAME:           BCPB
  SYSTEM NUMBER:         010000F3
  IMAGE NAME:            LP3
  NETWORK ADDRESS:       IBM390PS.R132
  IPL TOKEN:             C7FC41F7 C536C463
  DIAG INFO:             N/A
IXC111I LOGICAL PARTITION REMOTE CONNECTION INFORMATION 049
  CONNECTION STATUS:     CONNECTED
  SYSTEM NAME:           BCPI
  SYSTEM NUMBER:         020000F5
  IMAGE NAME:            LPB
  NETWORK ADDRESS:       IBM390PS.H87
  IPL TOKEN:             C80F0B49 87B47D86
  DIAG INFO:             N/A
IXC103I SYSTEM IDENTIFICATION INFORMATION 050
  CONNECTION STATUS:     CONNECTED
  SYSTEM NAME:           BCPG
  SYSTEM NUMBER:         030000F8
  IMAGE NAME:            LP8
  NODE DESCRIPTOR:       002097.IBM.02.000000062DBF
  PARTITION NUMBER:      08
  CPC ID:                00
  NETWORK ADDRESS:       IBM390PS.H87
  IPL TOKEN:             C80F5219 E1C92013
IXC104I SYSTEM STATUS DETECTION PARTITIONING PROTOCOL ELIGIBILITY: 051
  SYSTEM CAN TARGET OTHER SYSTEMS.
  SYSTEM IS ELIGIBLE TO BE TARGETED BY OTHER SYSTEMS.
```

Programming 101 - BCPii Execution Environment

- Hardware levels *(BCPii targeted systems)*
  - *zEnterprise*
  - *z10 plus recommended microcode levels*
    - Close to full functionality
  - *z9 plus recommended microcode levels*
    - Some reduced functionality (no IPLTOKEN, reduced attributes, no temporary capacity options)
  - *Lower than z9*
    - Significantly reduced functionality (no HWICMD, reduced attributes)
- Software levels *(System(s) which BCPii runs on)*
  - *z/OS V1R10 + PTF, z/OS V1R11 and higher in base*
    -

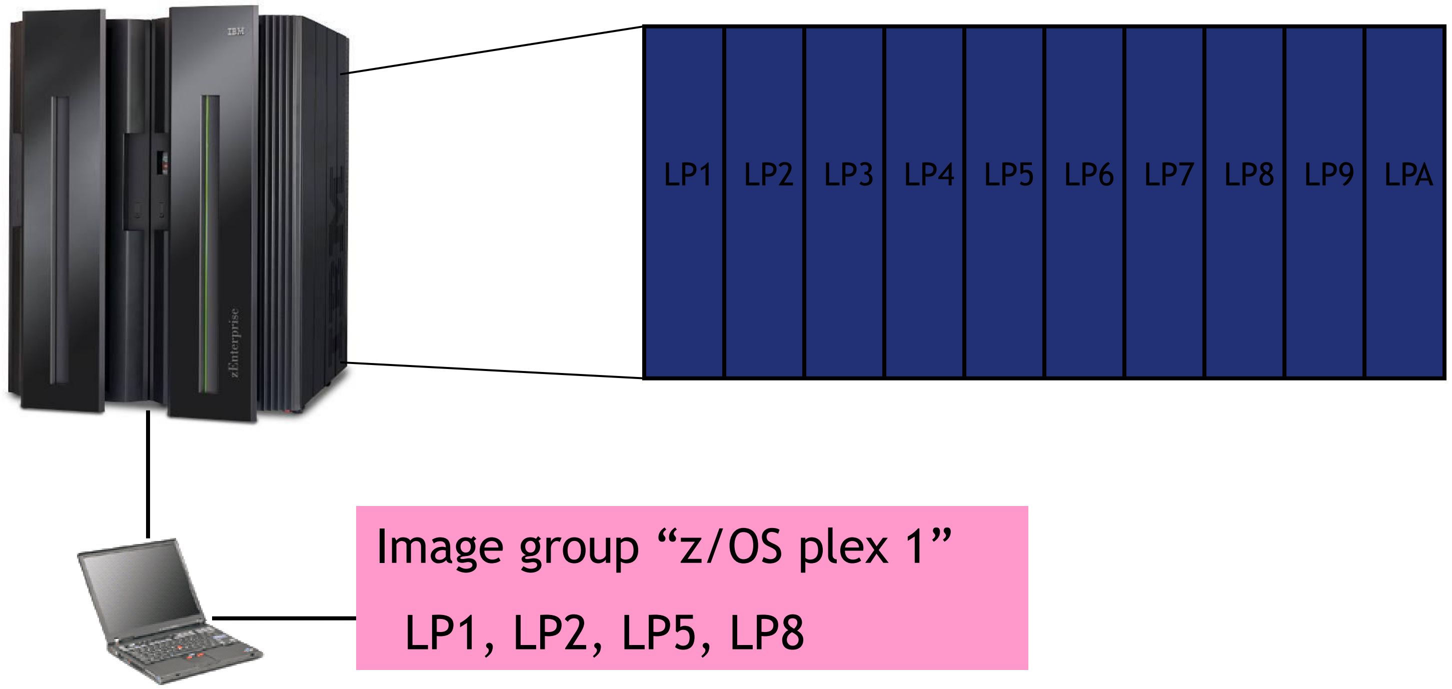
Programming 101 - BCPii Release Summary

- z/OS V1R10
  - –Base functions (no HWISET)
- z/OS V1R11
  - –HWISET
  - –Support for IPL Token / Query PSWs
  - –Activation profiles support
  - –Minor internal serviceability enhancements
- z/OS V1R12
  - –CTRACE enhancements
  - –Improved storage utilization and serviceability of BCPii transport code
  - –Additional CPC/Image attributes and commands

Programming 101 - BCPii Release Summary

- z/OS V1R13
  - –Support for user-defined image groups
  - –Additional CPC/Image attributes
  - –New STP commands

BCPii V1R13 Enhancements – User-defined image groups



- Use BCPii to work with this user-defined image group
  - List the image names in the group
  - Connect to, disconnect from the group
  - Query values associated with the group
  - Issue commands against all members in the group

BCPii V1R13 Enhancements – User–defined Image groups

- Important notes
  - Only z10 and higher hardware supported
  - Operating system, Activate with activation profile and System reset with IPLToken commands cannot be targeted to an image group
  - If the image group contains the local image the application is running on, only “non-suicide” commands will be permitted
  - Security is strictly enforced
    - **Application must have at least READ access to the CPC where the image groups reside in order to get list of image group names, to list the images residing in an image group, or to query attributes associated with the image group**
    - **Application must have at least CONTROL access to each of the images contained in the image group for all commands targeted to an image group. Access validation is done at the time of the HWICMD invocation**
  - BCPii handles if an image group definition changes under the covers after the application has already connected.

BCPii V1R13 Enhancements – User–defined Image groups

- Hardware Dependencies
  - z10 (Driver 79F)
    - MCL176 in the N24409 (SE-SYSTEM) EC stream
  - z196 (Driver 86E)
    - MCL220 in the N29802 (SE-SYSTEM) EC stream

BCPii V1R13 Enhancements – New Attributes

- HWIQUERY only (CPC attributes)
  - HWI_VERSION
    - Version of the Support Element console application associated with this CPC.
  - HWI_EC_MCL_INFO
    - Engineer code and microcode levels installed on this CPC.
  - HWI_LIST_IP_ADDRESSES
    - IP addresses associated with this CPC
  - HWI_AUTO_SWITCH_ENABLED
    - Flag indicating whether auto switch between the primary and the alternate support elements is enabled on this CPC

BCPii V1R13 Enhancements – New Attributes

- HWIQUERY and HWISET (Image attribute)
  - HWI_GROUP_PROFILE_CAPACITY
    - Workload unit capacity for the group profile associated with an image.

BCPii V1R13 Enhancements – New Attributes

- New HWICMD CmdTypes
  - HWI_CMD_SYSPLEX_TIME_SWAP_CTS
  - HWI_CMD_SYSPLEX_TIME_SET_STP_CONFIG
  - HWI_CMD_SYSPLEX_TIME_CHANGE_STP_ONLY_CTN
  - HWI_CMD_SYSPLEX_TIME_JOIN_STP_ONLY_CTN
  - HWI_CMD_SYSPLEX_TIME_LEAVE_ONLY_CTN

- Please exercise extreme caution when issuing any of these commands. Joining the STP-only CTN may result in a disabled wait for all images that are in a parallel sysplex on the target CPC.

Programming 101 - Programming Environment

- Services available in any address space
  - Program-authorized, and
  - SAF-authorized
- C and Assembler programming languages
  - REXX: FITS requirements MR0409106649 and MR0930096444 asking for BCPii System REXX support answered as “Accepted” by IBM on 1/24/11.
- z/OS UNIX callers can receive event notifications thru z/OS UNIX-only services utilizing the Common Event Adapter (CEA)

Programming 101 – Language Support

- Interface Definition Files (IDF, or include files) provided by BCPii:
  - C *(provided in SYS1.SIEAHDRV.H)*
    - HWICIC – Main BCPii include file
    - HWIZHAPI – Additional constant definitions include file
  - Assembler *(provided in SYS1.MACLIB)*
    - HWICIASM – Main BCPii include file
    - HWIC2ASM – Additional constant definitions include file

Programming 101 – Programming Environment

- Two ways to link your BCPii program:
  - –Use the linkable stub routine HWICSS from SYS1.CSSLIB to link-edit your object code.
  - –Use the LOAD macro to find the address of the BCPii callable service at run time and then CALL the service

Programming 101 - Programming Environment - Samples

- BCPii sample programs *(provided in samplib)*:
  - –C sample written in Metal C:

    HWIXMCS1 provides an example of how to use all of the traditional BCPii APIs and how to construct a simple BCPii application.

    HWIXMCX1 provides a simple example of how a BCPii Event Notification Facility (ENF) exit could be coded to field various BCPii-registered events.

Programming 101 - Let’s Meet the APIs!

- Functions performed using BCPii APIs:
  - Obtain the System z topology of the current interconnected CPCs, Images (LPARs) and their associated capacity records, activations profiles and user-defined image groups
  - Query various CPC, image (LPAR), capacity record, activation profile and user-define image group information
  - Set various CPC, image(LPAR), and activation profile information
  - Issue commands against both the CPC and image to perform hardware and software-related functions
  - Listen for various hardware and software events which may take place on various CPC and images

Programming 101 - Let's Meet the APIs!

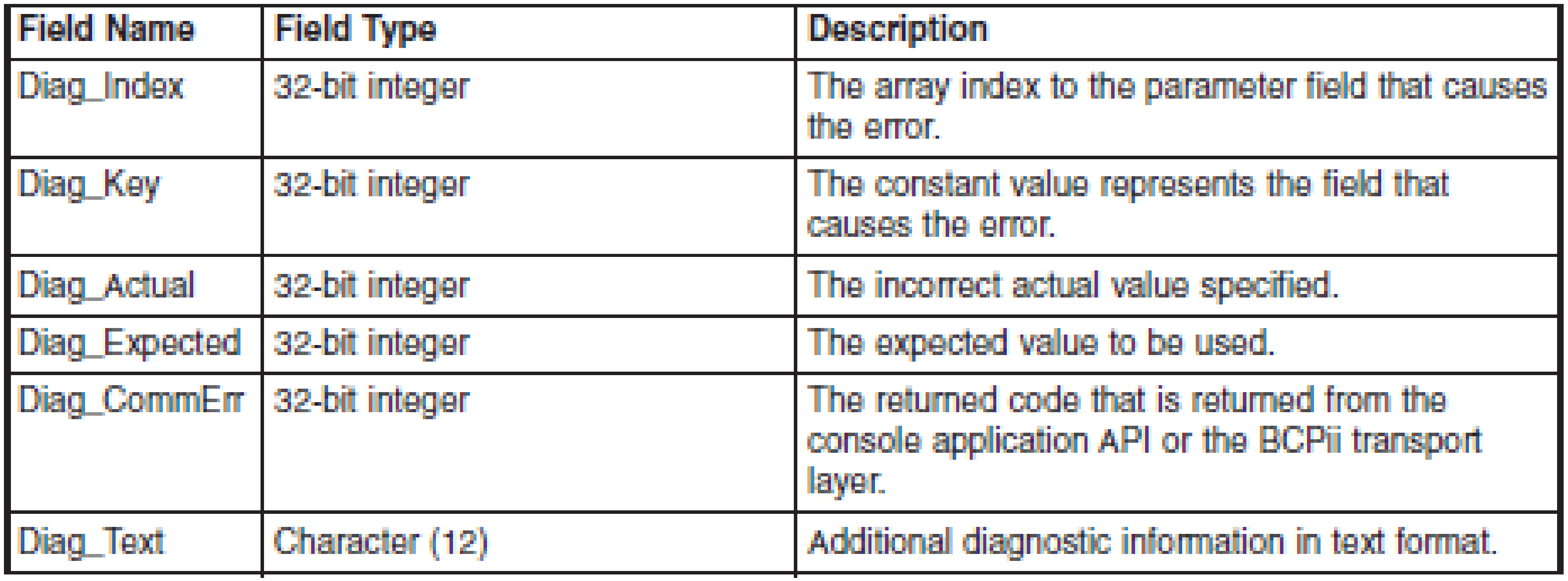
- Services available
  - HWILIST (BCPii List)
  - HWICONN (BCPii Connect)
  - HWIDISC (BCPii Disconnect)
  - HWIQUERY (BCPii Query)
  - HWISET (BCPii Set) – introduced in V1R11
  - HWICMD (BCPii Command)
  - HWIEVENT (BCPii Event (for non-z/OS Unix callers))
  - HwiBeginEventDelivery, HwiEndEventDelivery, HwiManageEvents, HwiGetEvent (for z/OS Unix callers)

Programming 101 - Let’s Meet the APIs!

- Services available
  - HWILIST (BCPii List)
  - HWICONN (BCPii Connect)
  - HWIDISC (BCPii Disconnect)
  - HWIQUERY (BCPii Query)
  - HWISET (BCPii Set) – introduced in V1R11
  - HWICMD (BCPii Command)
  - HWIEVENT (BCPii Event (for non-z/OS Unix callers))
  - HwiBeginEventDelivery, HwiEndEventDelivery, HwiManageEvents, HwiGetEvent (for z/OS Unix callers)

Programming 101 - Let’s Meet the APIs!

- All services pass back two groups of information used to determine the success of the request
  - Return code
    - 0 – Request completed successfully
  - DiagArea
    - Area that is filled in for certain non-environmental failures



Field Name	Field Type	Description
Diag_Index	32-bit integer	The array index to the parameter field that causes the error.
Diag_Key	32-bit integer	The constant value represents the field that causes the error.
Diag_Actual	32-bit integer	The incorrect actual value specified.
Diag_Expected	32-bit integer	The expected value to be used.
Diag_CommErr	32-bit integer	The returned code that is returned from the console application API or the BCPii transport layer.
Diag_Text	Character (12)	Additional diagnostic information in text format.

Programming 101 - Let's Meet the APIs!

- **HWILIST** - Retrieve HMC and BCPii configuration-related information
  - List CPCS
    - List the CPCs interconnected with the local CPC
  - List Images
    - List the images (LPARs) contained on an individual CPC or in user-defined imagegrp
  - List Capacity Records
    - List the capacity records contained on an individual CPC
  - List Events
    - List the events already registered on a particular BCPii connection
  - List Local CPC, List Local Image (available in V1R11)
    - Obtain the name of the CPC name or image (LPAR) name that the BCPii application is currently running on.
  - List Reset Activation Profiles, List Image A.P. and List Load A.P. *(available in V1R11 via APAR OA29638)*
    - List the currently defined activation profiles contained on a individual CPC
  - List User-defined Image Group Names
    - List the currently defined image group names contained on an individual CPC.

Programming 101 - Let’s Meet the APIs!

```
CALL HWILIST (
 ReturnCode
,ConnectToken
,ListType
,NumofDataItemsReturned
,AnswerArea_Ptr
,AnswerAreaLen
,DiagArea)
```

Programming 101 - Let’s Meet the APIs!

- **HWICONN** - Establish a logical connection between the application and a:
  - Central processor complex (CPC),
  - CPC image (LPAR) on a particular CPC,
  - Capacity record on particular CPC
  - Activation Profiles
  - User-defined image groups
- Input:
  - Connection type (above 3 types)
  - Connection name (CPC example: net1.cpc01)
  - Previous ConnectToken (if type is image, caprec, activation profile, or user-defined image group)
- Output:
  - ConnectToken used on subsequent BCPii calls.

Programming 101 - Let’s Meet the APIs!

```
CALL HWICONN (
 ReturnCode
,InConnectToken
,OutConnectToken
,ConnectType
,ConnectTypeValue_Ptr
,DiagArea)
```

Programming 101 - Let's Meet the APIs!

- **HWIDISC** – Release a logical connection no longer needed
- Input:
  - ConnectToken
- Note: Connections are implicitly disconnected when a job completes associated with the BCPii application (JES or z/OS UNIX initiator) or when the address space has terminated

Programming 101 - Let’s Meet the APIs!

```
CALL HWIDISC  (
 ReturnCode
,ConnectToken
,DiagArea)
```

Programming 101 - Let’s Meet the APIs!

- **HWIQUERY** - Retrieve information about objects managed by the hardware management console (HMC)/support element related to:
  - Central processor complexes (CPCs),
  - CPC images (LPARs) on a particular CPC,
  - Capacity records on particular CPC
  - Activation Profiles (Reset, Image, or Load)
  - User-defined Image group properties
- Input:
  - ConnectToken (associated with one of the above)
  - List of attributes requested, data areas to store the return values)
- Output
  - Data returned

Programming 101 - Let's Meet the APIs!

- Examples of information you can query
  - CPC information
    - General information
      - **Name, serial, machine type, id, networking info**
    - Status information
      - **Operating status and other status values**
    - Capacity information
      - **Various CBU info, Capacity on Demand info, Processor configuration, including IFA, IFL, ICF, IIP**
    - Power savings information *(available on zEnterprise hardware only with APAR OA34001 on V1R10, V1R11 and V1R12)*
      - **Is power savings available?, current power savings mode, supported power saving modes available**
  - Image information
    - General information
      - **Name, OS info**
    - Capacity information
      - **Defined capacity, Processor weights**

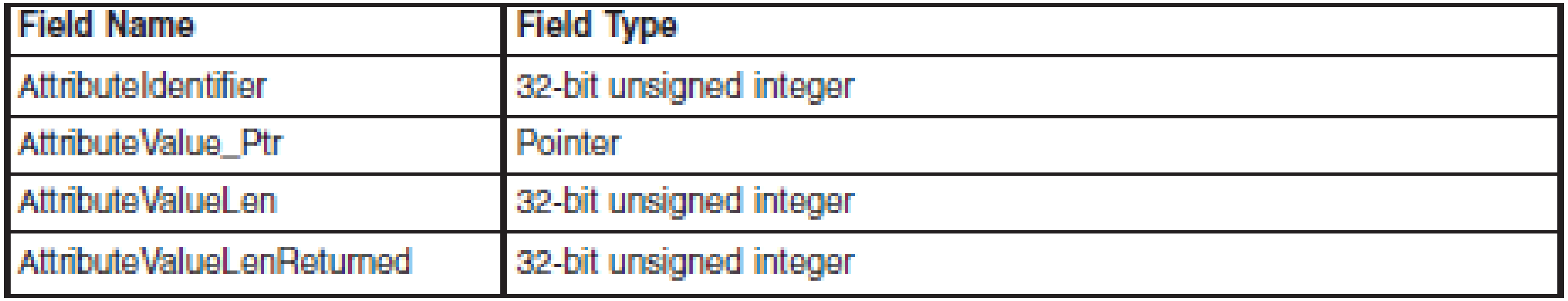
Programming 101 - Let's Meet the APIs!

- Examples of information you can query (continued):
  - Capacity record information
    - General information
      - **Name, Activation and expiration dates, activation days**
    - Status information
      - **Record status**
    - Capacity information
      - **The entire Capacity record**
  - Activation profile information
    - Most activation profiles values.

Programming 101 - Let’s Meet the APIs!

```
CALL HWIQUERY  (
 ReturnCode
,ConnectToken
,QueryParm_Ptr
,NumOfAttributes
,DiagArea)
```

Structure of one QueryParm:



Field Name	Field Type
AttributeIdentifier	32-bit unsigned integer
AttributeValue_Ptr	Pointer
AttributeValueLen	32-bit unsigned integer
AttributeValueLenReturned	32-bit unsigned integer

Programming 101 - Let’s Meet the APIs!

- HWISET *(available in V1R11)* – Change or set data for objects managed by the hardware management console (HMC)/support element related to:
  - Central processor complexes (CPCs),
  - CPC images (LPARs) on a particular CPC,
  - Activation Profiles *(available in V1R11 via APAR OA29638)*
- Input:
  - ConnectToken (associated with one of the above)
  - Attribute (object) to modify, the modified value, the value length
- Output
  - Return code

Programming 101 - Let’s Meet the APIs!

```
CALL HWISET  (
 ReturnCode
,ConnectToken
,SetType
,SetTypeValue_Ptr
,SetTypeValueLen
,DiagArea)
```

Programming 101 - Let’s Meet the APIs!

- Examples of information you can set
  - CPC information
    - Acceptable status values
    - Next Reset activation profile name
    - Processor Running Time
  - Image information
    - Various processor weights
  - Activation Profile Information *(available in V1R11 via APAR OA29638)*
    - Most activation profile values

Programming 101 - Let's Meet the APIs!

- **HWICMD** – Direct hardware/software commands to CPCs, images and user-defined image groups
- Input:
  - ConnectToken (associated with a CPC, image, or image group)
  - Command parameter structure (based on the type of command issued)
- Output
  - Synchronous return code
  - Asynchronous command completion event delivered to previously-registered event user when command finishes.
    - **For image commands targeted to an image group, one image event is returned for each image in the user-defined image group.**

Programming 101 - Let’s Meet the APIs!

- Examples of commands that can be issued:
  - –CPC commands
    - Activate, Deactivate an entire CPC
    - CBU request
      - **Activate or Undo**
    - On/Off Capacity on Demand request
      - **Activate or Undo**
    - Switch Power Savings Mode (*available on zEnterprise hardware only with APAR OA34001 on V1R10, V1R11 and V1R12)*
  - –Image commands
    - SysReset, SysReset with IPL Token *(V1R11)*
    - Load
    - Start, Stop all CPs
    - Add or remove temporary capacity
    - Issue operating system command

Programming 101 - Let's Meet the APIs!

```
CALL HWICMD  (
 ReturnCode
,ConnectToken
,CmdType
,CmdParm_Ptr
,DiagArea)
```

CmdParm_Ptr points to the command parameter list that is unique for each command. The data structure specified is defined in the IBM-supplied IDF files.

Programming 101 - Let’s Meet the APIs!

- **HWIEVENT (non-z/OS Unix callers) –** Register/Un-register an application and its connection to be notified for hardware and software events occurring on the connected CPC or image.
- Input:
  - ConnectToken (associated with a CPC or image)
  - Event action (Add or Delete)
  - Events for which an application wants to be notified
  - ENF exit to receive control when event arrives
- BCPii registers the user with ENF for this event(s) such that the ENF exit is driven only when the CPC and/or image name of the connector matches.

Programming 101 - Let's Meet the APIs!

- Examples of events that can be listened to:
  - Command completions
  - Status changes
  - Capacity changes
  - Disabled waits
  - Power mode changes *(available on zEnterprise hardware only with APAR OA34001 on V1R10, V1R11 and V1R12)*
  - BCPii status changes and communication errors

Programming 101 - Let’s Meet the APIs!

```
CALL HWIEVENT  (
 ReturnCode
,EventAction
,EventIDs
,EventExitMode
,EventExitAddr
,EventExitParm
,DiagArea)
```

Note: EventIDs is a 128 bit data structure is defined in the IBM-supplied IDF files as HWI_EVENTIDS_TYPE. Specify which events you wish to register with by turning on the appropriate bits. Make sure to fill in the beginning “eyecatcher” field with the constant value “HWIEVENTBLCK”

Programming 101 - Let’s Meet the APIs!

- **HwiBeginEventDelivery (z/OS Unix callers)** – begin delivery of event notifications.
- Input:
  - ConnectToken (associated with a CPC or image)
- Output:
  - DeliveryToken
    - To be used on HwiManageEvents service

Programming 101 - Let’s Meet the APIs!

- **HwiEndEventDelivery (z/OS Unix callers)** – End delivery of event notifications.
- Input:
  - DeliveryToken

Programming 101 - Let’s Meet the APIs!

- **HwiManageEvents (z/OS Unix callers)** – Registers / un-registers for a list of hardware/software events.
- Input:
  - ConnectToken
  - DeliveryToken
  - Event action (Add or Delete)
  - Events to be registered/unregistered

Programming 101 - Let's Meet the APIs!

- **HwiGetEvent (z/OS Unix callers)** – Retrieve outstanding event notifications.
- Input:
  - DeliveryToken
  - Buffer
    - Where the ENF68 event data is to be returned
  - Timeout
    - How much time to wait for an event to occur
- Output:
  - ENF68 Event Data in supplied buffer

Programming 101 - Simple Programming Example

- Application contains calls like this:
  - **HWILIST** (ListCPCs)
  - For each CPC name returned above:
    - **HWICONN** (CPC name (input), CPCConnectToken(output))
    - **HWIQUERY** (CPCConnectToken (input), QueryParms (HWI_CBUTESTAR))
    - **HWILIST** (CPCConnectToken(input), ListImages)
    - For each image returned above:
      - **HWICONN** (CPCConnectToken(input), Image name (input), ImageConnectToken(output))
      - **HWIQUERY** (ImageConnectToken(input), QueryParms(HWI_OSNAME,HWI_OSTYPE,HWI_OSLEVEL,HWI_SYSPLEX,HWI_DEFCAP)
      - **HWIDISC**(ImageConnectToken)
    - **HWIDISC** (CPCConnectToken)

Programming 101 - Simple Programming Example (HWILIST)

```
rc = -1;
numofCPCs = -1;
listtype = HWI_LIST_CPCS;
memset(List_of_CPCs, 0x00, sizeof(List_of_CPCs));
answerarealen = sizeof(List_of_CPCs);
answerarea_ptr = &answerarea[0];

hwilist(&rc,CPCoutconnecttoken,listtype,&numofCPCs,
        &answerarea_ptr,answerarealen,&diagarea);

if ( rc == 0 )
{
 printf("HWILIST for CPC: RC = %x\n",rc);
 printf("NumOfDataItem:   %d\n",numofCPCs);

 CPC_AnswerArea_into_Array(answerarea, List_of_CPCs, numofCPCs);
```

Programming 101 - Simple Programming Example (HWICONN)

```
while( i < numofCPCs )
{
 printf("CPC %d: %s\n",i+1,&List_of_CPCs[i].element);
 /* CPC ********************************************/
 /* HWICONN the CPC */
 rc = -1;
 memset(CPCinconnecttoken, 0x00, sizeof(CPCinconnecttoken));
 strcpy(CPC_target,List_of_CPCs[i].element);
 CPCconnecttypevalue = &CPC_target[0];
 CPCconnecttype = 1;

 hwiconn(&rc,CPCinconnecttoken,&CPCoutconnecttoken,CPCconnecttype,
         &CPCconnecttypevalue,&diagarea);

 if ( rc == 0 )
 {
  printf("HWICONN on %s: RC = %x\n",&List_of_CPCs[i],rc);
```

Programming 101 - Simple Programming Example (HWIQUERY)

```
/* HWIQUERY */
/* Calling HWIQUERY, using the returned output connecttoken from
HWICONN, query for HWI_MMODEL */
printf("HWIQUERY for HWI_MMODEL\n");
rc = -1;
numofattributes = 1;
Queryparm[0].AttributeIdentifier = HWI_MMODEL;
Queryparm[0].AttributeValue_Ptr  = &HWI_MMODEL_value[0];
Queryparm[0].AttributeValueLen   = sizeof(HWI_MMODEL_value);
Queryparm[0].AttributeValueLenReturned = -1;
query_ptr = (char *)&queryparm[0];

hwiquery(&rc,CPCoutconnecttoken,(void **)&query_ptr
         ,numofattributes,&diagarea);

if ( rc == 0 )
{printf("HWIQUERY on %s: RC = %x\n",&List_of_CPCs[i],rc);
```

Programming 101 - Simple Programming Example (HWIEVENT)

```
memcpy(&eventExitParm,&userdata,sizeof(eventExitParm));
memset(&eventIDs,0,sizeof(eventIDs));
strcpy(eventIDs.Hwi_EventID_EyeCatcher,HWI_EVENTID_TEXT);
eventIDs.Hwi_Event_CmdResp = 1;
eventIDs.Hwi_Event_DisabledWait = 1;
eventAction = HWI_EVENT_ADD;
eventExitMode = HWI_EVENT_TASK;
eventExitAddr = 0;
__asm ( " LOAD EP=HWIXMCX1  " : "=r"(eventExitEP) :  );
eventExitAddr = (int)eventExitEP;

/* -------------- */
/* Call HWIEVENT  */
/* ---------------*/
hwievent(returncodePtr,
         connecttoken,
         eventAction,
         eventIDs,
         eventExitMode,
         eventExitAddr,
         &eventExitParm,
         &diagarea);
```

Programming 101 - Simple Programming Example (HWICMD)

```
/* ------------------------------------------------------------*/
/* Initialize the cmdParm to null.                             */
/* Note: An OS command string must be null-terminated.         */
/* ------------------------------------------------------------*/
memset(&cmdParm,0,sizeof(cmdParm));
cmdParm_Ptr = &cmdParm;
cmdParm.PriorityType = HWI_CMD_NONPRIORITY;
strcpy(cmdParm.OSCMDString,"D GRS");
cmdType = HWI_CMD_OSCMD;
HWI_DIAGAREA_TYPE  diagarea;
/* ------------ */
/* Call HWICMD  */
/* ------------ */
hwicmd(returncodePtr,
       connecttoken,
       cmdType,
       &cmdParm_Ptr,
       &diagarea);

if(*returncodePtr) print_diagarea(diagarea);
```

Programming 101 - Simple Programming Example (Event Exit)

```
typedef struct {
  HWIENF68 * ENFEventDataPtr;    /* Data for a specific BCPii event   */
  int    reserved1;
  int    ENFUserData;            /* Optional user-supplied data       */
  int    reserved2;
  int    reserved4;
  int    reserved5;
} ENFDATA_TYPE;

 int main(ENFDATA_TYPE ENFData)
{
switch (ENFData.ENFEventDataPtr->eventType)
  {
```

Programming 101 - Simple Programming Example (Event Exit)

```
/* ------------------------------------------------------------ */
   /* Event Type Hardware Event                                    */
   /* ------------------------------------------------------------ */
   case HWIENF68_EVENTTYPE_HWEVENT:
     /* ---------------------------------------------------------- */
     /* Check for a specific event subtype                         */
     /* ---------------------------------------------------------- */
     switch (ENFData.ENFEventDataPtr->eventSubType)
       {
       /* -------------------------------------------------------- */
       /* Handle Command Response event subtype.                   */
       /* -------------------------------------------------------- */
       case HWIENF68_HWEVENT_CMDRESP:
```

Programming 101 - Simple Programming Example (Event Exit)

```
          /* ------------------------------------------------------ */
          /* Look at the eventdata fields for this eventSubType      */
          /* which are mapped by the HWIENF68_CMDRESP_T              */
          /* structure.                                              */
          /* ------------------------------------------------------ */
          /* Check whether this command response is one for         */
          /* which you are waiting by comparing the connect         */
          /* token to a saved connect token.                        */
*/
          /* ------------------------------------------------------ */
          /* Validate a saved connect token.                        */
          /* ------------------------------------------------------
  {if(savedConnectToken == ENFData.ENFEventDataPtr->
                        eventData.CmdResp.connectToken)
*/
if(1)                           /* If connect token matches */
   . . . . . .
```

BCPii Programming Kicked Up a Notch – Dynamic Config Handling

- If your program is long running, new entities may be added to the system, old ones deleted, or names changed. For example, activation profiles.
- Consider registering for HWIENF68_HWEVENT_NAMECHG, HWIENF68_HWEVENT_OBJCREATE, and HWIENF68_HWEVENT_OBJDESTROY events
- When activation profile is added, for example, the OBJCREATE event will be driven, with the name of the object, the object type (e.g. Image actprof), and the name of the CPC (cpcName field in the HWIENF68) all in the event data passed to the exit.
- Your program takes the appropriate action.
- Note: Please apply PTF UA62449 for V1R11, UA62450 for V1R12, or UA62451 for V1R13.

BCPii Programming Kicked Up a Notch – Handling Communication Outages

- It is possible that BCPii could lose connectivity to the CPC that you are connected to and waiting for events on.
- BCPii can tell you about these outages, allowing your program to take the appropriate actions:
  - HWIENF68_HWCOMMERROR_TEMP (ENF Qual 02010001)
    - BCPii detected that it lost connectivity momentarily with the target CPC but has regained connectivity
    - BCPii application should take the appropriate action
  - HWIENF68_HWCOMMERROR_PERM (ENF Qual 02010002)
    - BCPii detected that it lost connectivity and cannot regain connectivity to the CPC at the moment.
    - All HWIEVENT and HWICMD API requests are not processed, no events are delivered
    - BCPii application should wait for the HWIENF68_HWCOMMERROR_AVAIL event

BCPii Programming Kicked Up a Notch – Handling Network Outages

- HWIENF68_HWCOMMERROR_AVAIL (ENF Qual 02010003)
  - BCPii has established communications with the registered CPC (either at first connectivity to that CPC or when communications have resumed to that CPC

- An application has a choice of how to register for all hardware communication events:
  - Via HWIEVENT ADD service (EventIDs parameter value Hwi_Event_HwCommError)
  - Via ENFREQ LISTEN macro invocation specifying to listen for ENF68,
    - ENFREQ ACTION=LISTEN
    - CODE=ENFPC068, QUAL=02010000

BCPii Programming Kicked Up a Notch – Handling BCPii Outages

- While very rare, it is possible for the BCPii address space to go away unexpectedly.
  - BCPii signals an ENF68 with a QUAL of 01000002 when the address space becomes active
  - BCPii signals an ENF68 with a QUAL of 01000001 when the address space becomes unavailable
- An application should register itself with ENF to listen for these two events from occurring so it can take the appropriate actions
  - When BCPii goes down, all connections are lost. The application should throw away all recollection of connect tokens
  - When BCPii come back up, all connections should be reestablished.

BCPii Programming Kicked Up a Notch – Debugging Programming Errors

- API Return Codes and Diag Area
- CTRACE
  - –BCPii cuts CTRACE records using SYSBCPII CTRACE comp
  - –Default CTRACE CTIHWI00 parmlib member shipped
  - –Two CTRACE options:
    - Min
    - All
  - –Dump is taken whenever CTRACE is turned off
- Symptom Records
  - –Limited first failure data capture for select problems
- Support Element Tracing

BCPii Publications

- z/OS MVS Programming: Callable Services for High-Level Languages
  - Primary BCPii documentation including:
    - Installation instructions
    - BCPii API documentation
- z/OS MVS Programming: Authorized Assembler Services Reference, Volume 2 (EDT-IXG)
  - BCPii's ENF68 documentation
- z/OS MVS System Commands
  - START HWISTART and STOP HWIBCPII commands
- z/OS MVS Diagnosis: Tools and Service Aids
  - BCPii's CTRACE documentation
- z/OS MVS Initialization and Tuning Reference
  - Miscellaneous documentation
- z/OS MVS System Codes
  - BCPii abend '042'x documentation